

ENTERPRISE GUIDE

The Enterprise Guide to AI-Assisted Development

AI coding agents now run in the cloud, in parallel and on their own. The harder question is whether the organization can use them in a way that holds up under pressure.

Three problems slow engineering teams down. One operating model fixes them. One team has installed it across 100+ teams.

40%

velocity gain in the first month

95%

team-wide adoption after the workshop

100+

teams trained worldwide

PROBLEM

The cost of getting AI adoption wrong

AI coding tools are already inside your engineering organization, and agents now open pull requests on their own. The question is not whether people use them. It is whether the way they use them is making you faster or creating new problems nobody is measuring yet.

Review becomes the bottleneck

Engineers produce more code, faster. Reviewers cannot keep up. Senior people spend their time checking AI-generated output instead of making architecture decisions.

Quality drifts without anyone noticing

AI-generated code looks clean before it is actually sound. Teams confuse polish with judgment and only see the damage later.

Adoption is uneven, so the gains are uneven

Research shows senior engineers get 22% faster with AI tools while juniors get only 4% faster. Without one standard, leverage stays concentrated.

These are not tool problems. They are management problems. The organizations that fix them do not buy a better tool. They install a better operating model.

FRAMEWORK

Delegate. Review. Own.

Everything we teach comes from one observation: the teams that get reliable, repeatable value from AI-assisted development are the ones that make three boundaries explicit.

DELEGATE	REVIEW	OWN
Hand off repetitive tasks entirely: boilerplate, migrations, tests, documentation, and logs. Trust but verify.	The agent proposes, you decide: bug fixes, refactoring, optimization, edge cases, and test quality.	Core architecture, security decisions, business logic, standards, and risk acceptance stay human.

MOMENT	WHAT HAPPENS
Before the task	Name the task and the boundary it sits in: delegate, review, or own.
During the task	The agent drafts and proposes. It never decides what ships.
After the task	A person reads the evidence (the diff, the tests, the result) and takes ownership of the outcome.

Once the whole team shares these boundaries, agentic development stops being an individual experiment and becomes a team capability.

APPLICATION

Applying the framework across development phases

Delegate-Review-Own applies across the whole software lifecycle: where agents help, what engineers verify, and what stays owned.

PHASE	DELEGATE	REVIEW	OWN
Plan	Summarize inputs, draft options, map dependencies	Check assumptions and scope	Vision, trade-offs, approval
Design	Generate component skeletons	Check accessibility and performance	System design, architecture
Build	Infrastructure code, configurations, boilerplate	Audit security and cost implications	Architecture decisions
Test	Generate test suites and draft coverage	Reject shallow coverage	Test strategy, data approach
Review	Style, pattern, and security checks	Verify intent and audit drift	Standards, risk acceptance
Deploy	Run deployments, monitor logs	Audit configurations	Release decisions, incidents

When the team disagrees on a cell, that cell should default to Own until standards are clarified.

MATURITY

Where your team stands today

Teams move through five levels. Most start at one or two. Each level asks for one more thing to be shared, written down, or automated.

LEVEL	WHAT IT LOOKS LIKE	THE NEXT MOVE
L1 Assisted	AI helps a person with autocomplete and chat. Team habits have not changed.	Let an agent do a whole task, and review the evidence.
L2 Delegated	An agent does the task. A person checks the diff, the tests, and the result.	Share one way of working across the team.
L3 Systematic	Shared instructions, context files, and review checks live in the repository.	Let checks, not people, decide what may merge.
L4 Governed	Automated checks decide what merges. Every action is logged and measured.	Let the system improve itself, with people setting the direction.
L5 Self-improving	Context and tests update from production. People set direction and audit.	Keep auditing. Keep ownership.

Find your level in two minutes. Six questions, free: harness.institute/matrix

PRACTICES

The six operational practices

These practices come directly from the workshop curriculum and have been tested across more than 100 teams.

PRACTICE 1**Establish documentation for AI**

Agents get better when conventions, testing expectations, and business constraints are written down in shared rules files and skills, kept in the repository and reviewed like code.

PRACTICE 2**Write plans before prompting**

Strong teams shape the work first, then ask the agent to help execute it.

PRACTICE 3**Reset early when output goes wrong**

AI-generated code is cheap to produce and expensive to repair. Restart weak drafts sooner.

PRACTICE 4**Decompose work into small units**

One prompt should map to one independently reviewable task, even when cloud agents run several at once.

PRACTICE 5**Use verification checklists and gates**

Review gets easier when security, permissions, and test quality are explicit. Make automated review a merge gate, and open new tool connections read-only.

PRACTICE 6**Protect core engineering skills**

Architecture, debugging depth, prioritization, and judgment under uncertainty remain human.

Start with practices 1 and 2: write the rules down, then plan before you prompt.

WORKSHOP

What the workshop delivers

This guide explains the operating model. The workshop is where the team applies it to real work on the real codebase.

AREA	WHAT HAPPENS
Real workflows	We start with planning, implementation, review, QA, or release work that already carries pressure.
Your codebase	The team applies Delegate-Review-Own to its own codebase with AI coding agents, under NDA, with at least two trainers.
Carry-over	We leave behind review checks, planning patterns, handoff rules, and usage standards.
Execution-first delivery	Applied implementation, not abstract AI theory or generic templates disconnected from your stack.
Format	Typically two days, ten to thirty people, on your own repository. USD 1,000 per person per day, first Pulse month included.

**Hands-on application over passive observation.
The team leaves with patterns and controls it can
use immediately.**

NEXT STEP

Choose your next step

Start with the free check. Then pick the path that fits your team, or you.

OPTION	FOR	PRICE
Two-minute check	Any team	Free, at harness.institute/matrix
Harness Pulse	Teams that want to stay current	USD 1,500 a month, up to ten engineers
Team workshop	Teams that want to learn on their own code	USD 1,000 per person per day
Live cohort	One developer	USD 1,000 or USD 2,500, January 2027

Rogier Muller is CTO at BlueMonks Group and ambassador for Anthropic. **Vasilis Tsoilis** is co-founder and partner at Cognitiv+. They teach engineering and regulated teams to adopt AI without losing control of quality, review, and risk.

The goal is simple: better execution, cleaner review, and a team-wide standard that holds up under delivery pressure.